Hier haben wir eine weitere Anwendung auf das Max 2-Sat Problem (vgl. S. 10.52): Man hat hier eine Formel wie $F = (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge x_4 \wedge (\neg x_1 \vee x_4) \wedge \ldots$, und die Aufgabe ist es eine Belegung der Variablen zu finden, die die maximal mögliche #Klauseln von F wahr macht. Für Max-E2-Sat Formeln gibt es Belegung, die $\geq \frac{3}{4}$ aller Klauseln wahr macht, aber auch das hilft nicht, das Maximum zu finden. Unser Ziel wird es sein einen Algorithmus mit Zeit $O(c^m)$, $c < 2$, $m = \#$ Variablen anzugeben.

Aufgabe: Man mache sich klar, wozu irgendwelche backtracking Verbesserungen oder auch lokale Suche hilft nicht erfolgversprechend sind.

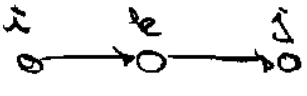
Sei $A = (a_{i,j})_{\substack{1 \le i \le m \\ 1 \le j \le m}}$ eine

G ungerichtet

Adjazenzmatrix von G. Wir betrachten die Matrix $A \cdot A = (b_{i,j})_{\substack{1 \le i \le m, \\ 1 \le j \le m}}$.

Was bedeutet doch der Eintrag $b_{i,j}$?

Es ist



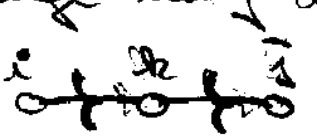
$$b_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot a_{k,j}$$

$$b_{i,i} = \sum a_{i,k} \cdot a_{k,i}$$

$= \#$ Wege

$= \#$ Grad (i)

$= \#$ der Wege von G der Art



- # Wege $i \rightsquigarrow j$ mit Länge $= 2$.

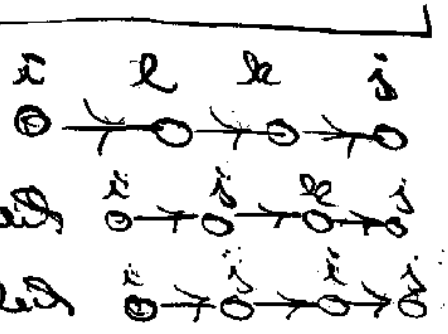
Bilden wir $A \cdot (A \cdot A) = (c_{i,j})_{\substack{1 \le i \le n \\ 1 \le j \le n}}$

Was bedeutet das $c_{i,j}$?

$$c_{i,j} = \sum_{l=1}^{n} a_{i,l} \cdot b_{l,j}$$

$$= \sum_{l=1}^{n} a_{i,l} \cdot \left(\sum_{k=1}^{n} a_{l,k} \cdot a_{k,j} \right)$$

$$= \sum_{l=1}^{n} \sum_{k=1}^{n} a_{i,l} \cdot a_{l,k} \cdot a_{k,j}$$



$= \#$ Wege $i \to j$ mit

3 Schritten (d.h. Länge 3)

Uns interessiert der Fall $i = j$

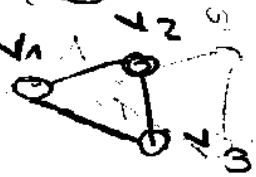
Dann $l \neq i$, $l \neq j$, $k \neq i$, $k \neq j$

(da keine Kanten wie $i \to i$)

Also ist

$c_{i,i} = \#$ Wege $i \to k \to l \to i$

$k \neq i, l \neq i, k \neq l$

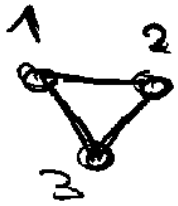
<u>Bezeichnung:</u> Ein Dreieck in G ist eine Mge. von 3 Kanten v_1v_2, v_2v_3, v_3v_1 mit  in G. $\square$

Ein [illegible]

Wo [illegible]

[illegible] Dann

11.61

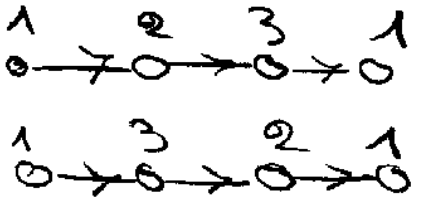
Ein Dreieck, etwa 

indiziert

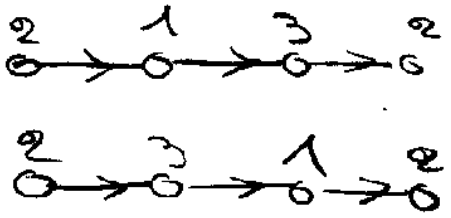
genau 6 Wege, die auf der

Diagonalen von $A \cdot (A \cdot A)$ gezählt

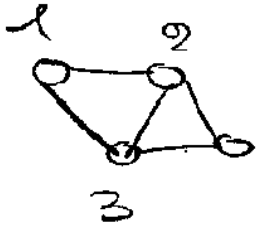
werden:

$c_{1,1}$ 

(2 ungerichtete)

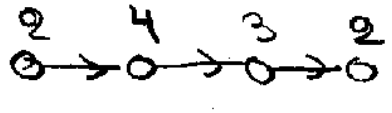
$c_{2,2}$ 

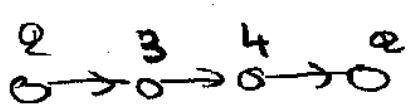
$c_{3,3}$ analog.

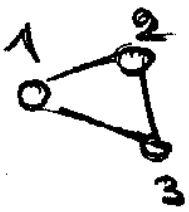
Haben wir 

,

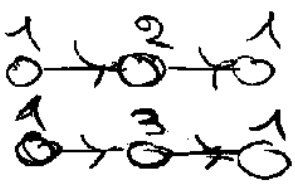
dann in $c_{2,2}$



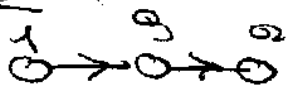


Beispiel: $G =$ 

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$



$$A \cdot A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$



$$(A \cdot A)\, A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2 & \cdots \\ \cdots & 2 \\ & & 2 \end{pmatrix}$$

Also: tatsächlich $\sum c_{i,i} = 6$

bei einem Dreieck! Und

$$\sum c_{i,i} = 6 \cdot \#\text{Dreiecke}.$$

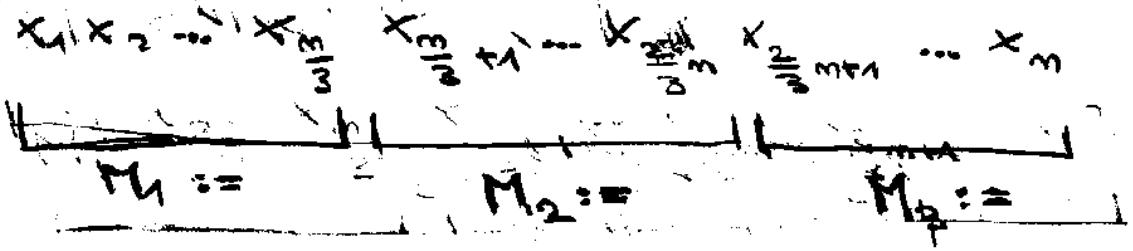
$\Rightarrow$ # Dreiecke zählen in $O(n^{\log_2 7})$

Beachte: Einfach alle Dreiermengen durchprobieren $\mathcal{O}(n^3)$ und $\Omega(n^3)$, da

$$\#\text{Dreiermengen} = \binom{n}{3} = \frac{n(n-2)(n-3)}{3!}$$

Wir übersetzen jetzt Max-2-Sat in das Problem Dreiecke zu zählen!

Entscheidend ist (die einmalige (nicht rekursive) Einteilung der $n \in \mathbb{N}$ Variablen von n Variablen in 3 gleichgroße Mengen:



M. 54

Zu Beachten folgende Beobachtung:

Jede Zweierklausel ist genau von einem der folgenden Typen:

Typ 1 Ganz in M_1 oder ein Literal aus M_1, eins aus M_2

Typ 2 Ganz in M_2 oder ein Literal aus M_2, eins aus M_3.

Typ 3 Ganz in M_3 oder ein Lit aus M_3, eins aus M_1.

Damit vereinfacht sich unser Problem:

BEISPIEL

$x_1\ x_2 \quad x_3\ x_4 \quad x_5\ x_6$

$M_1 \qquad M_2 \qquad M_3$

$$\#\cdot \underbrace{(x_1 \vee x_3)}_{\Gamma 1} \wedge \underbrace{(\neg x_1 \vee x_6)}_{\Gamma 3} \wedge \underbrace{(x_1 \vee x_2)}_{\Gamma 1} \wedge \underbrace{(x_1 \vee \neg x_4)}_{\Gamma 1} \wedge \underbrace{(x_4 \vee x_5)}_{\Gamma 2}$$

Definition ((i_1, i_2, i_3)-2-Sat Problem)

Eingabe: 2-KNF $F = C_1 \wedge \dots \wedge C_m$ über Variablen $x_1, \dots, x_n$.

Frage: Gibt es Belegung $a \in \{0,1\}^n$, so daß für $1 \le j \le 3$

$$i_j = |\{k \mid C_k \text{ vom Typ } j\}|$$

Alternativ: # derartige Belegungen.

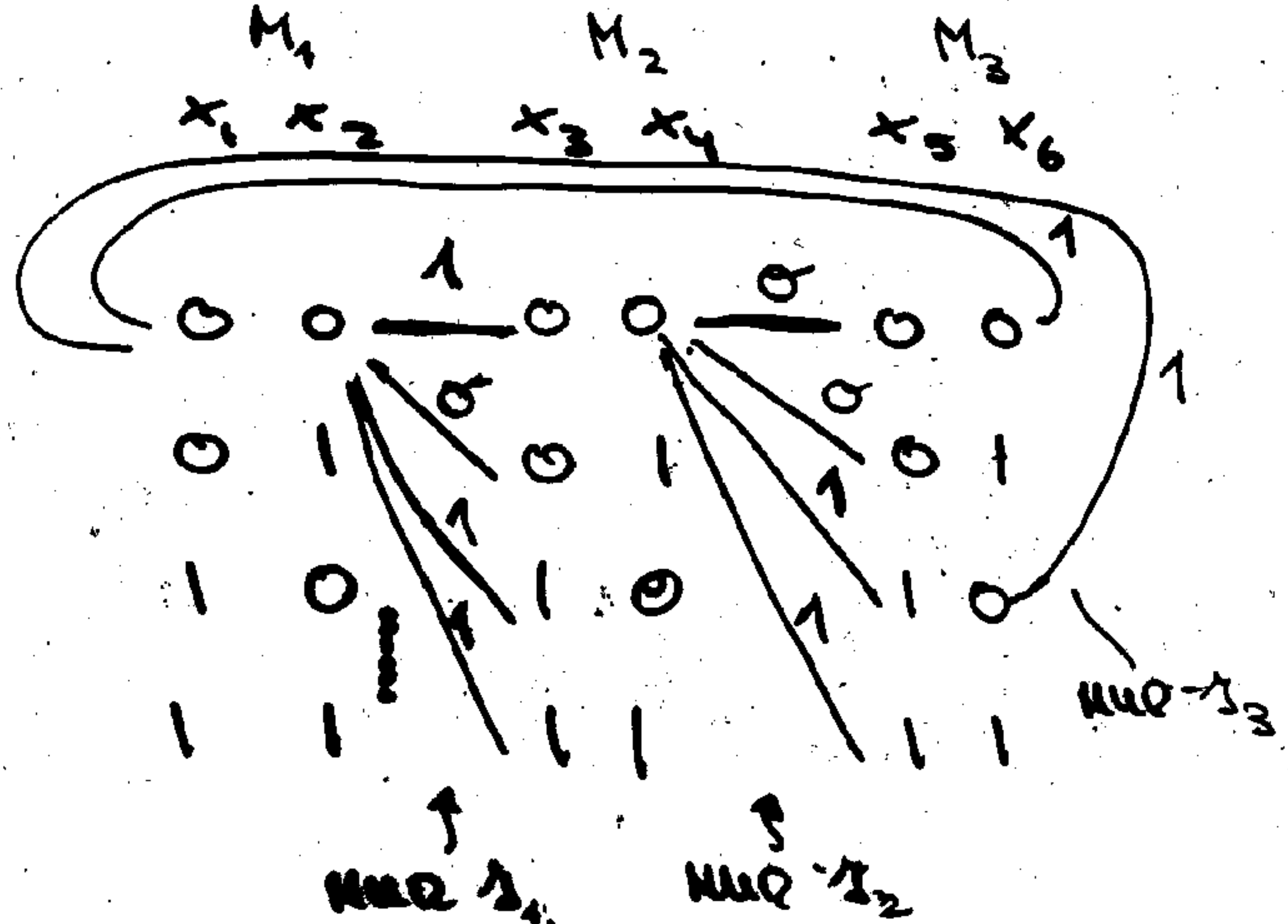
F erfüllbar $\Leftrightarrow$ Es gibt (i_1, i_2, i_3) mit $\sum i_j = m$ so daß pos. Antwort für das (i_1, i_2, i_3)-2-Sat Problem.

Klar in $O(m \cdot 2^n)$ lösbar. Aber es geht besser.

BEISPIEL ZUM ALGORITHMUS

DAS (1,1,1)-2-SAT PROBLEM.

$$F = (x_1 \vee x_3) \wedge (\neg x_1 \vee x_6) \wedge (x_1 \vee x_2) \wedge (x_1 \vee \neg x_4) \wedge (x_4 \vee x_5)$$



$(1_1, 1_2, 1_3)$ - 2-SAT

DREIECKE SUCHEN.

Algorithmus $((i_1, i_2, i_3)\text{-2-Sat})$

Eingabe: 2-Sat $F = C_1 \wedge \dots \wedge C_m$,
Tripel (i_1, i_2, i_3),
über $x_1, \dots, x_n$.

Ausgabe: # Belegungen a so daß

$i_j = \{k \mid a(C_k) = 1 \text{ und } C_k \text{ vom Typ } j\}$.

1. $V_1 = \{(b_1 \dots b_{\frac{n}{3}})_1 \mid b_i \in \{0,1\}\}$

$V_2 = \{(b_1 \dots b_{\frac{n}{3}})_2 \mid b_i \in \{0,1\}\}$

$V_3 = \{(b_1 \dots b_{\frac{n}{3}})_3 \mid b_i \in \{0,1\}\}$

// $V = V_1 \cup V_2 \cup V_3$, kombiniere

// $|V| = 3 \cdot 2^{n/3}$

2. Für jedes $b, c \in \{0,1\}^{\frac{n}{3}}$ // Typ 1

3. Berechne (b,c) der Belegung a

$a(x_{bm} x_1 \dots x_{\frac{n}{3}m}) = \dots$

4. $q := |\{x \mid C_x \text{ vom Typ } 1, a(C_x) = 1\}|$;

5. If $q = i_1$

Kante $(b)_{i_1} - (c)_2$ einbauen. }

6. for each $(c,d) \in \{0,1\}^{\frac{1}{3}m}$ // Typ 2

7 (c,d) als Belegung a von

$x_{\frac{1}{3}m+1} \dots x_m$

8. $q := |\{x \mid C_x \text{ vom Typ } 2, a(C_x) = 1\}|$

9. If $q = i_2$

Kante $(c)_2 - (d)_3$ }

10. Ebenso for each $d,b \in \{0,1\}^{\frac{1}{3}m}$ // Typ 3.

Analog.

11. $A := $ Adjazenzmatrix // $\Theta(4^{\frac{m}{3}})$
// Einfügen.

12. $B := A \cdot A \cdot A$

13. Anzahl „1" in $\frac{1}{6} \sum_{i,i} b_{i,i}$

(M.50)

Laufzeit

10 & 11: $O(4^{\frac{m}{3}} \cdot m)$

P.o. Paar ≠ durchgeführt und zählen.

12. $O(2^{\frac{m}{3}} \cdot \overbrace{\log_2 7}^{<3})$.

13. $O(2^{\frac{m}{3}})$

Also: $O(2^{\frac{m}{3}\log_2 7})$, da

$\log_{\frac{1}{2}} 7 > 2$ und $4^{\frac{1}{3}} = 2^{\frac{2}{3}}$.

M. Ca

ALGORITHMUS (MAX 2-SAT)

EINGABE: $F = C_1 \wedge \dots \wedge C_M$ 2-KNF

ÜBER $X_1, \dots, X_N$

AUSGABE: #BELEGUNGEN b

FÜR DIE GILT

$$|\{x \mid b(C_x) = 1\}|$$

$$= \max\{|\{x \mid a(C_x) = 1\}| \mid a \in \{0,1\}^N\}$$

1. FOR $1 \le I_1, I_2, I_3 \le M$ {

2. $E[I_1, I_2, I_3] :=$ (I_1, I_2, I_3)-2-SAT MIT $\neq$

3. SUCHE DEN EINTRAG

$E[I_1, I_2, I_3] > 0$ MIT

DEM $I_1 + I_2 + I_3$ MAXIMAL.

$O(2^{\frac{N}{3}\log_2 7} \cdot N^4)$. DIES IST NUR!

Als nächstes wird das Problem
des Auswahl (selection) behandelt.

Gegeben ist ein

array A[1,...,n] of „vergleichbarer Datentyp"

und eine Zahl i, $1 \le i \le n$. Gesucht
ist das i'te größte Element
(= das Element, das bei einer Sortierung
von A[1], ..., A[n] an der i'ten Stelle
steht.)

- Ist $i = 1$, $i = n$, dann klar
  in Linearzeit $O(n)$.
- Ebenso für $i \le k$, k Konstante
  unabhängig von n in $O(n)$
  ($O(k \cdot n) = O(n)$ wenn k
  konstant)
- Aber ist bei $i = \lceil \frac{n}{2} \rceil$ (Sortieren braucht $O(n \cdot \log n)$.

Tatsächlich führt divide- and conquer zu $O(n)$ für alle i. Insgesamt ein Algorithmus überlinear mit Quicksort und konsequenterweise $O(n^2)$ im worst case.

```
Select(l,m,i){ // linker Rand, rechter
                // Rand, i, 1≤i≤m.
1. if l=m; { Ausgabe „A[l]"; return;}
2. Wähle A[j] für ein gewähltes j
                // Pivotelement
3. B1 := die Elemente von A[l,...,m],
        die ≤ a sind
   b1 := die # dieser Elemente
   if i ≤ b1; {                // Hier b1 ≥ 1 da
      A[l,...,b1] := B1; // i ≥ 1.
      Select(l,b1,i); return
   }
```

4. $B_2 :=$ die Elemente von A, die $> a$ sind;

$b_2 :=$ die # dieser Elemente;

$A[1, .., b_2] := B_2$;

Select $(1, b_2, i - b_1)$ // $i - b_1 \geq 1$ weil
// $i \leq b_1 + b_2 \leq b_2$;
// da $i \leq b_1 + b_2 = m$
// $b_2 < m$, da $b_1 \geq 1$. □

Korrektheit: Induktion über m:

$m = 1$, dann $i = 1$, da $1 \leq i \leq 1$, dann ✓.

$m = 2$, dann etwa $i = 2$, dann

wenn $A[1] = A[2]$ wird $b_1 = 2$

⇒ Endlosschleife. Aus wenn

$A[1] < A[2]$ und $a = A[2]$.

⇒ Man muß das Pivotelement vertauschen (vgl. Quicksort)

Also müssen wir 3. ändern.

3. $B_1 :=$ die Elemente von $A \leq a$;

$b_1 :=$ die # dieser Elemente;

$A[1,\dots,b_1] := B_1$;

→ if $b_1 = i$; ? Ausgabe „a"; return ?

Select $(1, b_1 - 1, i)$

Dann kann 1. auch entfallen.

Also noch einmal zur Korrektheit:

$n = 1$, dann ✓ (auch ohne 1.)

$n = 2$, $A[1] = A[2]$, $i = 2$, $j = 2$

dann $b_1 = 2$, $b_1 = i$ ✓

Falls hier $i = 1$, dann rekursiv

mit A' unverändert Select(1,1,1).

Jetzt beliebiges m. Ind.Vor für
$m' < m$ und alle i', $1 \leq i' \leq m'$.

~~Falls~~ $b_1 = m$, dann falls $i = m$ $\checkmark$,
falls $i < m$, dann $\checkmark$ nach Ind.-Vor.

~~Falls~~ $b_1 < m$, dann falls $i = b_1$ $\checkmark$,
falls $i < b_1$, dann $b_1 > 1$ dann
$select(1, b_1 - 1, i)$ terminiert nach Ind.-Vor $\checkmark$,
auch wg. Setzung von A. Ist
nun aber $i > b_1$, dann $i = b_1 + (i - b_1)$
und $select(1, b_2, i - b_1)$ terminiert,
Ind.-Vor $\checkmark$ auch wg. Setzung von A.
(Bem. $b_2 < m$, da $b_1 \geq 1$ und $b_1 + b_2 = m$.)

Komplexität.

$T(m) :=$ worst-case Zeit für
$select(1, m, i)$, wobei
$A[1..m]$ und i beliebig.

Sei

$T(n)$ = Zeit von Select(A, m, i_0),
wobei $A_0[1, \ldots, m]$ zugrundeliegt
und eine feste Wahl von j
(etwa $j = 1$).

Dann gilt für ein Konstantes d, gezeigt

$$T(n) \leq d \cdot n + T(n-1)$$

oder

$$T(n) \leq d \cdot n + T(n-2)$$

oder

$$T(n) \leq d \cdot n + T(n-3)$$

$$\vdots$$

oder

$$T(n) \leq d \cdot n + T(2)$$

oder

$$T(n) \leq d \cdot n + \text{// Zeit für } b_1 \leq \ldots \leq b_n$$
// Zeit für Ausgabe
// in d.

Also

$$T(n) \leq d \cdot n + \max\{T(n-i) \mid 1 \leq i \leq n\}$$

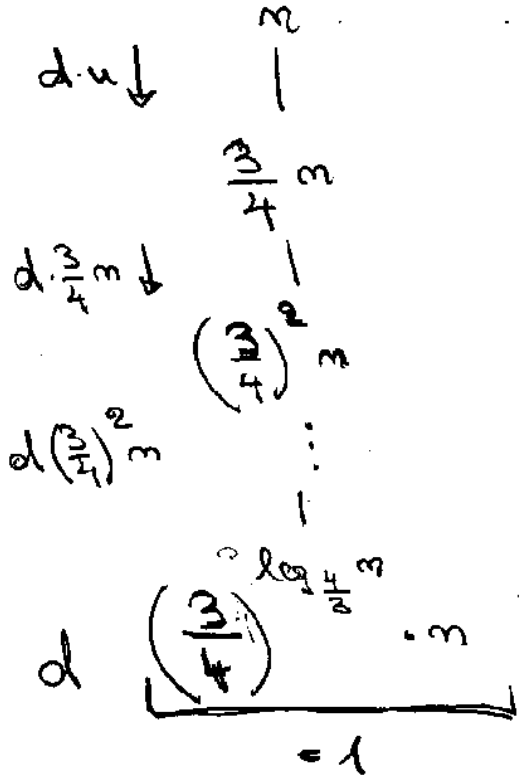
$$= d \cdot \left(\sum_{i=0}^{n-1} (n-i)\right) + T(0) \qquad T(1) = d$$

$$= \Theta(n^2)$$ ↯ dass gilt's auch nicht.

Nehmen wir einmal an

A würde immer um $\frac{1}{4}$ kleiner,

dann betrachtet man:



Tiefe des Baums: $\log_{\frac{4}{3}} n$

$T(n)$ = Zeit beim Aufteilbaum

$$T(n) = \sum_{i=0}^{\log_{\frac{4}{3}} n} \underbrace{\left(\frac{1}{\frac{4}{3}}\right)^i}_{<1} \cdot d \cdot n$$

$$\leq \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i \cdot dn = \frac{1}{1-\frac{3}{4}} \cdot d \cdot n$$

konvergiert — geometrische Reihe

$$= \frac{1}{\frac{1}{4}} d \cdot n = O(n)$$

Also linear und besser kann es bei beliebigem i nicht sein!

Ziel: Bestimmung des Pivotelements $a = A[j]$ derart, daß immer ein linearer Anteil abgespalten wird. Also a nicht im kleinsten Viertel (Fünftel, ...) und a nicht

im größten Viertel (Fünftel, ...)
wenn A geordnet wäre.

Idee: Wir gestalten uns dazu
mehrere rekursive Aufrufe,
sind Select(1, m, i) und zwar
vom Typ

Select(1, 'c...', -), Select('1, 'm', -),

wobei $c + c' \leq 1$. Das führt
bei linearem Zusatzaufwand
für das Divide und Conquer
auf

T(m) := worst-case Zeit von
Select(1, m, -)

Es

(11.21)

$$T(n) \leq d \cdot n + T(\overbrace{\lceil cn \rceil}^{<n}) + T(\overbrace{\lceil c'n \rceil}^{<n})$$

Sei nun f hinreichend groß, also konstant, dann impliziert

$$T(cn) \leq fcn, \quad T(c'n) \leq fc'n,$$

da

$$T(n) \leq dn + fcn + fc'n$$

$$= dn + f(c+c')n$$

$$\leq fn,$$

sofern

$$d + f(c+c') \leq f$$

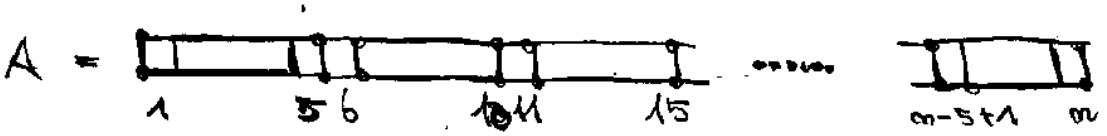
$$\Leftrightarrow \quad d \leq f(1-(c+c'))$$

$$\Leftrightarrow \quad \frac{d}{1-\underbrace{(c+c')}_{<1!}} \leq f.$$

Und f kann durch geeignet gewählt werden.

11.72

Also zur Bestimmung des Pivotelements a folgende Idee: Einteilung



Teile der Länge 5. Dann sortiere jeden Teil allein

$A[1-5], A[6-10], \dots, A[m-5+1, \dots, m]$

für sich. Dann nehme aus jedem Teil das mittlere Element (Median). Also bei sortierten Feldern:

$m_1 := A[3], m_2 := A[8], m_3 := A[13], \dots, m_{\frac{n}{5}} := A[n-2]$

$-5+3$

Es sind $\frac{n}{5}$ viele Elemente m_i.

Angenommen wir haben von diesen $\frac{n}{5}$ Elementen den Median, d.h. das a ist das

BEISPIEL

A = |1|3|7|10|5|2|8|13|27|21|3|5|6|7|11|

|105|1|22|11|12|18|3|16|42|1|

NACH SORTIERUNG DER FÜNFER

M_1= M_2=

A = |1|3|5|7|10|2|8|13|27|3|

M_3= M_4=

|3|5|6|7|11|1|11|12|37|105|

M_5=

|1|11|15|16|17|

↑

ARRAY DER 5'ER MEDIANE

M = 5 13 6 12 15

MEDIAN DER MEDIANE (REKURSION)

A = 12

PIVOTELEMENT FÜR DAS WEITERE.

$M_1, M_3, M_4 \leq A$

$M_2, M_5 > A$

das $\frac{m'}{2}$'te Element bei m' gerade,

das $\lceil \frac{m'}{2} \rceil$ Element bei m' ungerade.

Dann gilt, daß unter den Teilen $A[5 \cdot j \cdot m + 1, \dots, 5(j+1)m]$ von A noch drei, deren Median $m_{j+1} \leq a$ ist, mindestens $\frac{1}{2}$ Elemente $\leq a$ sind, nämlich m_{j+1} und die beiden kleinsten (die beiden größeren können auch noch $\leq a$ sein - egal). Weitere Elemente sind damit $\leq a$.

Also wir haben sicherlich

$\geq \frac{m'}{2} = \frac{3}{10} m$ Elemente $\leq a$ und

daraus $\geq \frac{3}{10} m$ Elemente $\geq a$.

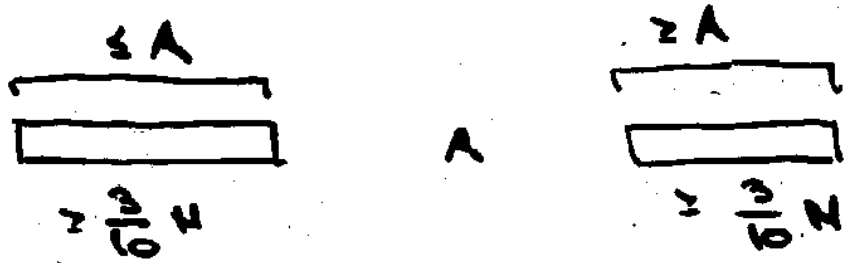
BEISPIEL

A - MEDIAN DES

FÜNFER MEDIANE

A SEI FOLGENDER MASSEN,

WENN DAS ARRAY SORTIERT IST:



KLEINES PROBLEMCHEN:

GLEICHE ELEMENTE

IM ARRAY!

ANNAHME: JEDES ELEMENT

NUR GENAU EINMAL

IM ARRAY.

11.76

Algorithmus (Lineares Select)

```
LS(A,i) {   // A = A[1,...,m], alle
            // Elemente verschieden, 1 ≤ i ≤ m,
            // m durch 10 teilbar.

1. if m ≤ 1000 then
       sortiere nach i'tes Element
       zurückgeben; return   // Ende der
   }                         // Rekursion

2. for 1 ≤ j ≤ m/5 do
       sortiere A[(j-1)·5+1,...,j·5],
       B[j] = A[(j-1)5+3]
                             // B[j] = Median der
                             // j'ten 5'er Teile
                             // von A.
   }

3. a := LS(B, 1/2 · 1/5 m)   // Bem.
   Pivotelement!             // m durch 10
                             // teilbar.
```

11.77

4. B_1 := die Elemente von A, die $\leq a$ sind;

b_1 := die # dieser Elemente;

if $b_1 = i - 1$; {

Rückgabe von „a“; end

return

}

5. if $i \leq b_1 - 1$ { // $b_1 - i \leq m$

Füllt B_1 ohnedies $b_1 - 1$ mit a auf

damit die Länge durch 10

teilbar wird

Rückgabe $L_i(B_1, i)$; return;

}

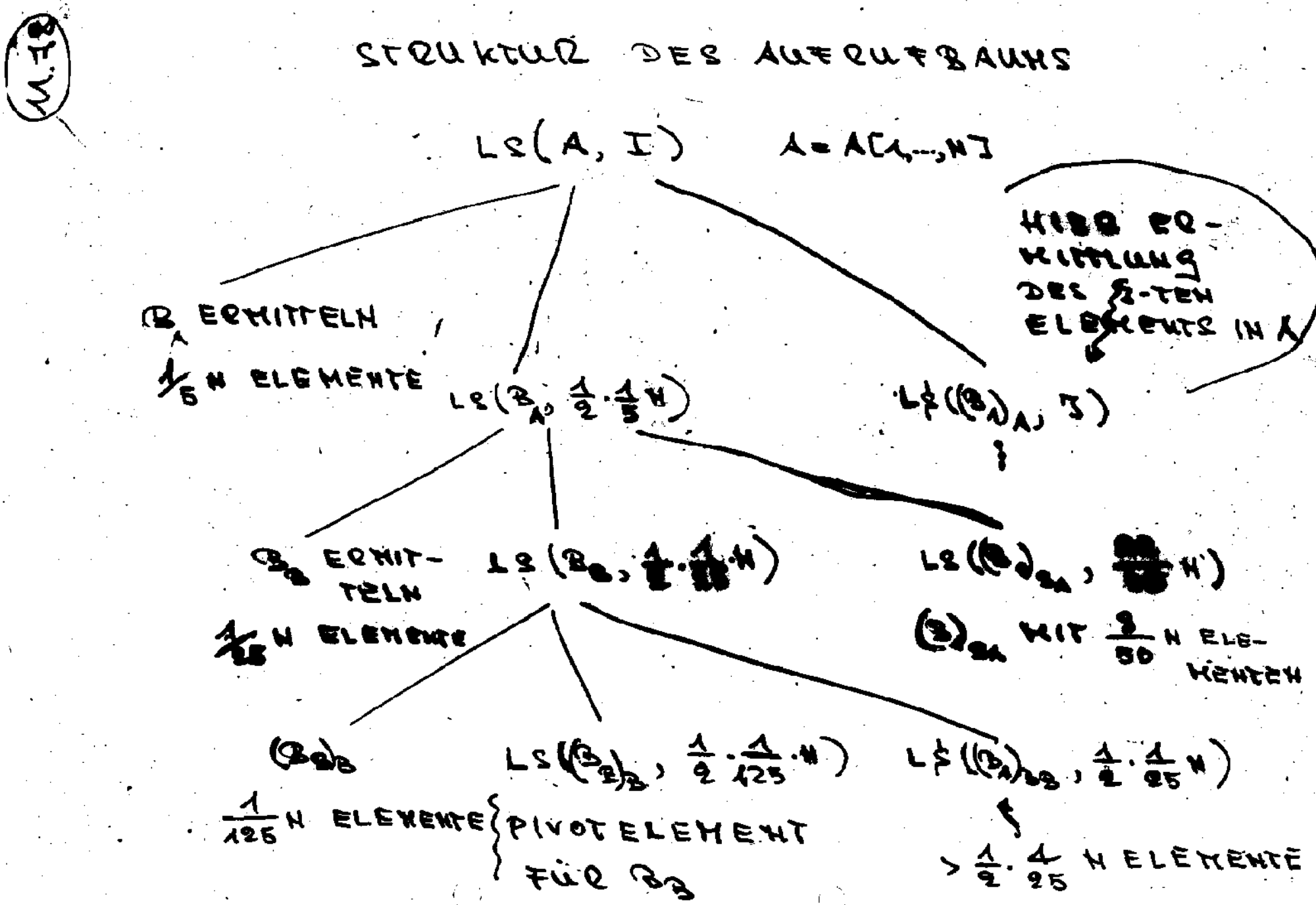
6. if $i > b_1$ {

dasselbe wie oben mit B_2;

Rückgabe von $L_i(B_2, i - b_1)$;

return; // $b_1 \geq 1$. Widerspr.

} // daß $a = A[1]$!


STRUKTUR DES AUFBAUMS
LS(A, I) A = A[1,...,N]
B_A ERMITTELN
$\frac{1}{5}$ N ELEMENTE
$LS(B_A, \frac{1}{2} \cdot \frac{1}{5} N)$
HIER BE-
STIMMUNG
DES $\frac{1}{2}$-TEN
ELEMENTS IN A
$LS((B_1)_A, I)$
B_{BA} ERMIT-
TELN
$\frac{1}{25}$ N ELEMENTE
$LS(B_{BA}, \frac{1}{2} \cdot \frac{1}{25} N)$
$LS((B_2)_{BA}, \frac{2}{50} N)$
$(B_2)_{BA}$ MIT $\frac{2}{50}$ N ELE-
MENTEN
$(B_{BA})_B$
$\frac{1}{125}$ N ELEMENTE
$LS((B_2)_B, \frac{1}{2} \cdot \frac{1}{125} N)$
PIVOTELEMENT
FÜR B_3
$LS((B_A)_{BA}, \frac{1}{2} \cdot \frac{1}{25} N)$
$> \frac{1}{2} \cdot \frac{1}{25}$ N ELEMENTE

(11.78)

Nun ist $\frac{1}{5} + \frac{7}{10} = \frac{9}{10} < 1$

und also linear Laufzeit zu erwarten.

Für $n \leq 1000$ ist

$$T(n) \leq d$$

für eine geeignete Konstante d.

Für $n > 1000$ bekommen wir

$$T(n) \leq d \cdot n + T\left(\tfrac{1}{5}n + 5\right) + T\left(\tfrac{7}{10}n + 9\right).$$

(wg. Teilbarkeit durch 10.)

Median $+ B_1$ oder B_2 des Medians

Nun ist bei $n > 1000$

$$5 \leq \frac{1}{200} \cdot n \qquad 9 \leq \frac{1}{100} \cdot n$$

(M 20)

Also bei $n \geq 1000$

$$T(n) \leq d \cdot n + T\left(\left(\frac{1}{5} + \frac{1}{100}\right) n\right) + T\left(\left(\frac{7}{10} + \frac{1}{100}\right) n\right)$$

$$\leq d \cdot n + c \cdot \underbrace{\left(\frac{9}{10} + \frac{2}{100}\right)}_{<1} n$$

$$\leq c \cdot n$$

$$= O(n).$$

Auch wenn $\frac{1}{5} + \frac{1}{100} n \leq 1000$!

wobei $c \cdot \frac{8}{100} > d$.

also $c > \frac{100}{8} \cdot d > d$!